RELIABILITY ISSUES IN THE ARPA NETWORK

W. Crowther, J. McQuillan, and D. Walden

Bolt Beranek and Newman Inc.
Cambridge, Massachusetts

RELIABILITY ISSUES IN THE ARPA NETWORK*

W. Crowther, J. McQuillan, and D. Walden
Bolt Beranek and Newman Inc.
Cambridge, Massachusetts

Introduction

Since the inception of the ARPA Network[1] in 1969, we have been part of the group responsible for the development of that network's communications subnet. This role has provided us with a unique opportunity for study of the problems of network reliability and the effects of attempted improvements, particularly in the context of rapid network growth. The communications subnet of the ARPA Network consists of store-and-forward, packet-switching nodes (called Interface Message Processors or IMPs[2,3]) connected together by wide-band communications circuits, usually with at least two paths to each IMP. As both the IMPs and the circuits occasionally fail, a variety of techniques (some routine, some novel) have been employed to minimize the effects of component failures.

Our overall philosophy for this effort has been that the network should be fault-tolerant with respect to individual component errors, and that the IMPs themselves should be fault-tolerant with respect to local failures. Along with this concern, we feel that the program should provide as much diagnostic information as possible. Component failures are of several kinds: hardware or software; solid, intermittent, or one-time. As we will discuss in the following sections, our attention has shifted in the last few years from handling circuit errors and failures to handling more difficult problems in the IMPs themselves: first intermittent problems, and recently even solid failures of major components.

The Present IMP

The ways of reacting to detected component failures in the IMP so as to improve fault tolerance include, in broad terms, retrying or recovering the last operation, or ignoring the failure, which can sometimes be done without data loss and sometimes not. Most diagnostic actions include reporting the failure to the Network Control Center (NCC)[4]. When necessary, the system is automatically or manually restarted or reloaded, perhaps after a core dump has been taken.

The specific techniques we have instituted to improve network reliability may be briefly summarized as follows. The original IMP, delivered in 1969, had hardware checksums and software to detect errors on the circuits, to declare circuits "dead" if it finds them too unreliable (in which case traffic is routed by an alternate path), and to report to the source when the IMP detects a failure to get a message delivered. In 1971 software checksums were added to the routing update messages passed between IMPs in order to detect errors caused by memory failures. In 1972 the software began reporting "impossible" software conditions and some hardware error conditions to the NCC. Also, an optional error-controlled interface to the computers using the network was provided. In 1973 we added software checksums to all packets transmitted between IMPs. Packets in which errors are detected are sent to the NCC, thus detecting memory, bus, and interface failures and providing diagnostic information on them to the engineering staff. Network end-to-end packet checksums have been added, along with checksums on routing messages between the time of creation and actual transmission. The code which performs the routing computation is checksummed every time it is used, causing a system reload if the code is detected as broken. A mechanism has been developed which allows the NCC to verify for correctness IMP core in IMPs that are running. Finally, the software modules are being separated as much as possible so that any one can be stopped, started, or reloaded with minimal disruption to the system as a whole. The trend toward more and more mechanisms to improve reliability is consistent, and is consistently advancing on two fronts: first, an attempt to make the software more robust in the face of failures; and second, an attempt to improve the gathering of diagnostic information in the event of failure.

As a result of the above-mentioned techniques, the following statements about the network are generally true:

--Single or intermittent failures in circuits, modems, or interfaces can be handled without data loss.
--Solid failures in circuits, modems, or interfaces can be accepted because of the double connectivity property of the network.
--Single or intermittent failures in memories or processors are probably infrequent and do not have an especially deleterious effect on global network performance, although the specific IMP suffering the failure may have to be reloaded or restarted.
--Solid failures in memories or processors again do not affect global network performance too seriously but, of course, the IMP is useless until repaired.
--Solid software failures are relatively easy to spot and fix in a 24-hour, 7-day-a-week, 40-node network, and they do not represent a major problem.
--Subtle software problems are very difficult to catch, and we see no surefire way to guard against them. Further, we find it increasingly difficult to find subtle software bugs using conventional debugging techniques; a bug which occurs once a week in the network occurs once every six months in a single machine. (As an aside, we are now in the position in our test cell that we can frequently debug with more nodes than are proposed for many experimental networks.) More and more

*This work was supported by the Advanced Research Projects Agency under contracts DAHC15-69-C-0179 and F08606-73-6-0027.

difficult bugs are being found just by hard thought and by study of the listing. We do take the very useful precaution of having the program periodically reset, in the main loop, every data and control structure which has been too long in one state.

The New IMP

Almost two years ago our group embarked on the development of a new minicomputer/multiprocessor IMP[5] which is to provide increased nodal throughput, flexibility, and reliability. Realizing the potential reliability of this new IMP has been an important concern during the development of both the hardware and software for the machine.

The hardware is constructed to allow at least two copies of any component (be it interface, clock, memory, processor, bus, etc.), so that no single component failure stops the system. With more than two of some components (e.g., we think of a system with fourteen processors as typical), reliability will be further improved. In fact, because big systems are configured primarily in the interest of throughput, reliability can be thought of as coming almost free in these systems. Of course, small systems with less than two of some component are also possible, but their reliability is knowingly sacrificed for lower cost.

It is an important principle of our design (both hardware and software) that we are willing to accept system errors (with low probability) as long as total system failure is avoided. Because of this principle we are not forced to use the redundant components in an expensive majority logic (or similar) manner, but instead can have all components normally perform independent tasks for great total system power--as long as when one component fails, the remaining system components take over the function of the failed component, albeit at reduced total system throughput. The software is written in a manner such that any processor can be scheduled for any task, making it very natural for correctly operating processors to be applied to any pending task even though some processors may have failed.

It is with the software that hardware or software failures are detected and recovery is initiated. Recovery is much easier than detection. The main difficulty with recovery mechanisms is making them as "gentle" as possible; that is, problems should be cleaned up without requiring a restart of the whole system (for example, if a buffer "gets lost", the software should go find it, not just reinitialize the entire buffer storage). An interesting dichotomy exists with regard to recovery: recovery from the failure of a passive device (e.g., memory) can consist of merely avoiding the use of the component while awaiting its replacement; recovery from the failure of an active device (e.g., a processor) requires immediate amputation of the device from the remainder of the system.

A number of techniques are used in the more difficult task of detecting that a problem exists. The code and static data structures are continually checksummed and rechecksummed using a general purpose checksum-box. All dynamic data structures are continually checked for reasonableness and cleaned up if found too long in a given state; thus, the software cannot "hang" in "impossible" states since it is always stepped to a legal state eventually. In the case of software destruction, new code can be loaded from elsewhere in the network. Processors cannot stop by mistake since they have no halt instruction. The processors periodically check each other, and there is a capability for removing a failed processor or other component. This is done in a manner making it very unlikely that an "insane" processor amputates all the rest of the system. Further, some particularly dangerous corrective actions can only be performed when the software presents a correct password to the hardware, and no single processor has the complete password.

Construction of reliable software is simplified by two assumptions: we assume there are no purposefully malicious programs operating within the system; and we assume the software is not required to detect all theoretically possible problems, only those which, practically speaking, do happen.

When completed, the new minicomputer/multiprocessor IMP should significantly improve node reliability. This, combined with the global network techniques described above, will greatly increase overall network reliability, in particular as it enables robustness in the face of circuit failures.

Conclusion

In the early stages of the construction of the ARPA Network, we were most concerned with attaining good performance with reasonable simplicity; poor reliability was not thought to be a serious problem. An interesting lesson, then, is how inexorably we have been drawn to the use of mechanisms explicitly added for improved reliability as the network has grown larger. In whatever performance improvements we now undertake, considerations of reliability will continue to play a major role.

1. Roberts, L.G., and Wessler, B.D., "Computer Network Development to Achieve Resource Sharing," AFIPS Proceedings, SJCC, 1970.
2. Heart, F.E., et al, "The Interface Message Processor for the ARPA Computer Network," AFIPS Proceedings, SJCC, 1970.
3. McQuillan, J.M., et al, "Improvements in the Design and Performance of the ARPA Network," AFIPS Proceedings, FJCC, 1972.
4. McKenzie, A.A., et al, "The Network Control Center for the ARPA Network," IEEE Proceedings, ICCC, 1972.
5. Heart, F.E., et al, "A New Minicomputer/Multiprocessor for the ARPA Network," AFIPS Proceedings, NCC, 1973.